Session 133

AppleScript Overview

Chris Espinosa
Manager, Components and Scripting

Introduction

- Yearly update on the status of AppleScript
- Key changes in Mac OS X that affect your scripts and applications
- Things that we know will change in the future
- Not a lot of forward-looking promises—let's focus on shipping apps for Mac OS X !

What You'll Learn

- New features of AppleScript 1.6 on Mac OS X to take advantage of
- What to do about your scripts and scripting additions
- How to plan for changes expected in future versions

Thank You, Developers!

TECSoft AppleScript Training
Brookline Software WebMiner 2.0
PineHill Products SiteToolkit 1.5
Donagrandi Software RBdbMaker 1.2
Simon Kornblith's IntelliDate
St. Clair Software Screen Catcher 2.3.4
Dragon Systems Power Manager 2.1
David Bakkers AShare Helper 2.0
Snak 4.6
iView PhotoMover Version 2.0
DssW Power Manager 2
NuSpectra Multimedia SiteCam 6.0.5
Aladdin Systems StuffIt Deluxe 6.0.1
Walter Ian Kaye BBE FTP Open 1.1.2
Intelli Innovations P.O. Toolbox 1.3
Apple Airport 1.3
Acrux Software iBuild
AcutAbove Software ScriptStrip
Intelli Innovations Server Sentinel 2.0.2
Runtime Labs MacSQL Monitor 1.6b2
REAL Software REALbasic 3
Christian Vick SETI UNiT Manager 3.3
Sophisticated Circuits iDo Script Scheduler 1.1.1
Ekim Software Socket Sifter Pro 1.8
Power On Now Up-to-Date and Contact 4.0
Lemke Software GraphicConverter 4.0.2
John Vink PhotoPage 1.3 beta 2
MCF Software ListSTAR 2.1

Thank You, Developers!

St. Clair Software Default Folder 3.0.9
Apple QuickTime 5
Jack Jansen MacPython 2.0
Casady & Greene SoundJam MP 2.5.2
Vampire Software OpenScript
Vampire Software OpenWorld
Runtime Labs MacSQL Monitor 1.1.1
Microsoft Office Extras
Optima System PageSpinner 3.0 Beta 4
David Blache IRCmagic 1.0.4
Chilton Webb Personal Site Administrator
Murry Yar Multiple Users Actions 1.0.1
Intelli Innovations Package Tracker
Dragon Systems Software Power Manager
Doug Holton Applescript Plugin 1.0
Quandir Software DayDreamer
Chilton Webb AppleEvent DecoderRing
Style 1.7
T & B AppleWorks Enhancement Pack 1.0.3
Mikael Hansen AutoShare 4.2
Michel Pollet Folder Compare 2.0
Mike Bailey Socket Sifter
Qualcomm Eudora Internet Mail Server 3.0

Version Madness 2001

- AppleScript 1.5
  - Shipped in Mac OS X Public Beta
- AppleScript 1.5.5
  - Based on 1.5
  - Shipped in Mac OS 9.1
- AppleScript 1.6
  - Shipped in Mac OS X version 10.0
  - Software Update available for Mac OS 9.1
  - Mac OS 9 and Mac OS X versions "substantially similar"

New in 1.5 through 1.6

- AEHelpers (née AEGizmos) in Carbon
- **keyAppHandledCoercion** flag for double ‘as’
- Unicode class support
- Support for packaged apps and additions
- Support for data-fork and ‘typeless’ script files
- Different methodology for dispatching events to scripting additions (Mac OS X only)

Bugs Fixed in 1.5 and 1.5.5

- **middle item of {}** crashed
- **item 0 of {1, 2}** didn't fail
- **info for** choked on files >2GB in size
- **typeChar** to **typeUnicodeText** assumed MacRoman
- Large seed values produced similar initial random nums
- 536870911 + 1 = –536870921
- **mount volume** didn't look at Keychain

Bugs Fixed in 1.6

- Read/Write Commands were broken in 1.5.5
- Null characters displayed as "\" or garbage
- Unicode operations crashed
- Standard Additions required Security Library
- Occasional inaccurate "out of memory" errors
- Scripts compiled on Mac OS X can't find peer apps on Mac OS 9

Bugs Known in 1.6

- **info for** fails on UFS/NFS volumes
- **typeUnicodeText** to **typeChar** coercion is broken
- **mount volume** doesn't work with 3rd-party UAMs
- **read as list using delimiters** is broken
- Launch-sends sometimes timeout

New Features

- AEHelpers (in Carbon Apple event manager only)
- Support for handling the double-‘as’ problem
- Unicode class support

AEHelpers

- Many developers depend on AEGizmos
- Couldn’t use old code libraries with Carbon
- Code dependent on now-opaque AE structure
- Now built in to Apple event manager on Mac OS X and CarbonLib 1.1 on Mac OS 9

AEHelpers: Build and Print

- Construct an event or descriptor from a formatted text string

  AEBuildDesc
  AEBuildParameters
  AEBuildAppleEvent

- Dump an event or descriptor to text

  AEPrintDescToHandle

AEBuild Syntax: Basic Types

- Constants: **3, “Untitled”, «4170706C», $3F216C0A$**
- Four-character codes: **psbr, 'WWWΩ'**
- Coercions: **real(1), type(cobj), 'utxt'(\"Untitled\")**
- Lists: **['TEXT', 'BOBO']**
- Records: **STXT{TEXT: "Hello", styl:«»}**
- Substitution: **@** (AEDesc), **type(@)** (long + ptr), **@@** (handle)

AEBuild: Building a Descriptor

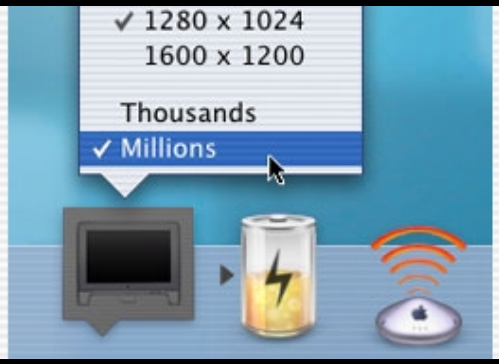
Every line of document “name”

```
err = AEBuildDesc(
    &desc,
    &buildErr,                    // for debugging only
     "'obj '({   want:type(clin),                          \
                 from:'obj '({ want:type(docu),            \
                               from:null,                  \
                                 form:name,                \
                                 seld:TEXT(@)              \
                               }),                         \
                 form:enum(indx),                          \
                 seld:abso('all ')                         \
            })",
    docName);
```

AEBuild: Building Events

```
err = AEBuildAppleEvent(
    'aevt', 'odoc',
    typeApplSignature,
    &finderSignature, sizeof(finderSignature),
    kAutoGenerateReturnID, kAnyTransactionID,
    &event,
    &buildErr,
     "'----':alis(@@)",
        itemToOpenAliasHandle
    );
```

Demo



AEHelpers: AEStream

- Progressively build a nested descriptor

AEStreamOpen
AEStreamClose
AEStreamOpenDesc
AEStreamWriteData
AEStreamCloseDesc
AEStreamWriteDesc
AEStreamWriteAEDesc
AEStreamOpenList
AEStreamCloseList

AEStreamOpenRecord
AEStreamSetRecordType
AEStreamCloseRecord
AEStreamWriteKeyDesc
AEStreamOpenKeyDesc
AEStreamWriteKey
AEStreamCreateEvent
AEStreamOpenEvent
AEStreamOptionalParam

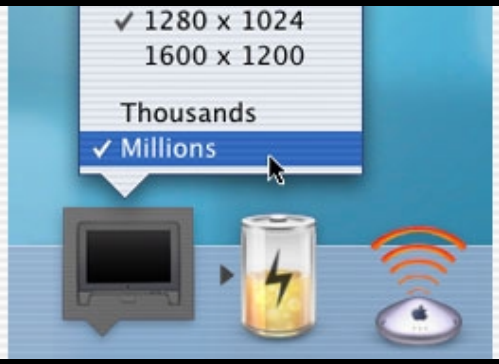
AEStreams Example

```
AEStreamRef s;
s = AEStreamOpen();
AEStreamOpenRecord(s, typeObjectSpecifier);
AEStreamWriteKeyDesc(s, keyDesiredType,
    typeType, &cLineCode, sizeof(OSType));
AEStreamWriteKey(s, keyAEContainer);
AEStreamOpenRecord(s, typeObjectSpecifier);
AEStreamWriteKeyDesc(s, keyDesiredType,
    typeType, &cDocCode, sizeof(OSType));
AEStreamWriteKeyDesc(s, keyAEContainer,
    typeType, &cNullCode, sizeof(OSType));
AEStreamWriteKeyDesc(s, keyAEKeyForm,
    typeType, &formNameCode, sizeof(OSType));
```

AEStreams Example (Cont.)

```
AEStreamWriteKeyDesc(s, keyAEKeyData,
    typeChar, &docName[1], docName[0]);
AEStreamCloseRecord(s);
AEStreamWriteKeyDesc(s, keyAEKeyForm,
    typeEnumerated, &formAbsolutePosition,
    sizeof(OSType));
AEStreamWriteKeyDesc(s, keyAEKeyData,
    typeAbsoluteOrdinal, &kAEAll, sizeof(OSType));
AEStreamCloseRecord(s);
AEDesc d;
AEStreamClose(s, &d);
```

Demo



AEHelpers

- What this means to developers
  - You can port AEGizmos code to Carbon despite the opaque AEDescs
  - Apple event debugging code is easier to write with a print facility built-in

New Features

- AEHelpers
- Support for handling the double-‘as’ problem
- Unicode class support

The “double-‘as’” Problem

- AppleScript supports the ‘as’ term two ways:
  - A binary operator, e.g.,
    "Hello" as Unicode text
  - A parameter to an Apple event, e.g.,
    get name of item 1 as Unicode text
- Your app gets a chance to look at the desired type to perhaps affect what value is returned
- AppleScript tries to coerce the result anyway!

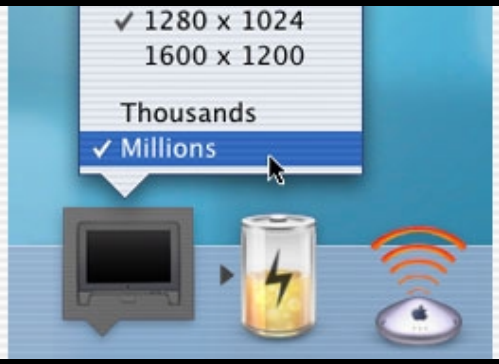
The “double-‘as’” Problem

- Result: **get every item as alias**
  - AppleScript got an error: can’t coerce
    {alias "Macintosh HD:Desktop Folder:foo", alias "bar" of desktop} to an alias
  - Or an improper coercion
    "Name 1Name 2Name 3Name 4"
- Especially troublesome for plural keyforms
- Forces scripters to write unnecessary loops

The “double-‘as’” Solution

- If you ignore the ‘as’ parameter, do nothing
  - AppleScript will continue to coerce the result
  - Scripts will be unaffected
- If you consider the ‘as’ parameter, provide **keyAppHandledCoercion** in the reply
  - AppleScript will not coerce the result
  - Script workarounds will continue to work
  - New scripts can avoid the workaround

Demo



The “double-‘as’” Solution

- What this means to developers
  - You can choose who does the coercion
- What this means to scripters
  - Your current scripts should work unchanged
  - You can remove workarounds for the problem when developers update

New Features

- AEHelpers
- Support for handling the double-‘as’ problem
- Unicode class support

Unicode Class Support

- Finder 10.0 now returns all file names as **typeUnicodeText**
- AppleScript 1.4.3 had coercion-only support
- 1.5 and 1.5.5 added (somewhat broken) full Unicode class support
- 1.6 fixed full Unicode class support
  - But broke the Apple event **utxt** to **TEXT** coercion!

Unicode Class Support

Elements

- character
- word
- paragraph
- text item

Properties

- length

Coercions

- list, record, string, styled text, number

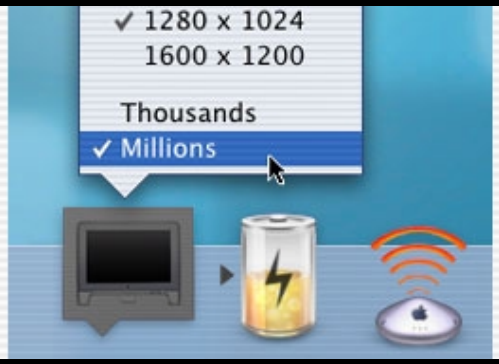
Operators

- &
- =, ≠, <, ≤, >, ≥
- contains, is contained by

Keyforms

- absolute position
- first, last, middle, some, every range

Demo



Unicode Class Support

- What this means to developers
  - Start returning appropriate values as **typeUnicodeText** (file names, etc.)
  - Scripts that work with strings will function normally
- What this means to scripters
  - You'll have access to multilingual text

Differences in Mac OS X

- Scripting Additions Folder Locations
- Packages
- Extensions

Scripting Additions Folders

- Follow the Locales model for FindFolder on Mac OS X
- /System/Library/ScriptingAdditions is for OS built-ins and is locked
- /Library/ScriptingAdditions is for machine-installed additions and has Admin access
- ~/Library/ScriptingAdditions is for user-installed additions and is open to each user

Scripting Additions Code Model

- Scripting Additions for Mac OS 9 *must* be InterfaceLib
- Scripting Additions for Mac OS X *must* be Carbon
  - May be CFM or mach-o
  - May be flat or packaged
- It's not worthwhile to build a package with both
  - You'll need separate copies for Classic and Mac OS X anyway

Scripting Additions Access

- Mac OS 9 and Classic behave as always
- Mac OS X scripts can use Classic additions only by targeting a running Mac OS 9 application
  - Launcher is a good target
- Most Mac OS X applications don't load additions

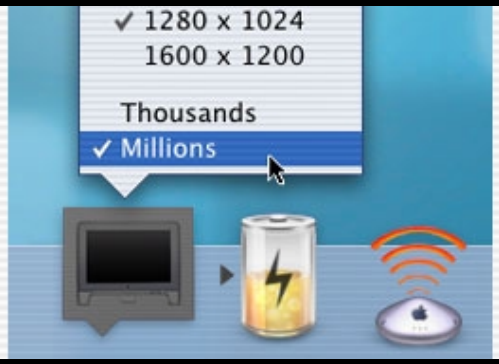
Scripting Additions Access

```
set Launcher to "Launcher"
set StartupDisk to (path to startup disk) as string
launch application (StartupDisk & "System
   Folder:Control Panels:" & Launcher)
   tell application Launcher
      using terms from application "Macintosh
                HD:System Folder:Scripting
            Additions:MonitorDepth"
      get monitor depth
   end using terms from
end tell
```

Scripting Additions Access

- On Mac OS X, scripting additions aren't loaded at startup time
- Most applications won't load scripting additions
  - Only OSA clients get them automatically
- Scripting addition events not handled by the target application will be redirected to the application that's running the script
- That application is guaranteed to have additions

Demo



Scripting Additions

- What this means to developers
  - Port your scripting additions to Carbon
  - Release both Mac OS 9 and Mac OS X versions
- What this means to scripters
  - Move Mac OS X addition calls out of Tell blocks
  - Enclose calls to Mac OS 9 additions in Tell blocks targeting a Mac OS 9 application

Differences in Mac OS X

- Scripting Additions Folder Locations
- Packages
- Extensions

Packages

- Most Mac OS X apps will be folders, not files
- **info for** will report applications are directories
- **path to** will return the path to the folder, not the actual executable application
- We'll make sure packages are treated as files wherever applicable

Differences in Mac OS X

- Scripting Additions Folder Locations
- Packages
- Paths and Extensions

Paths and Extensions

- Files and aliases still use Mac-style colon-delimited paths
- We probably won't ever support slash-delimited paths directly
  - Instead, we'll support file URLs widely

Paths and Extensions

- The Finder sometimes hides file extensions (like "**.app**")
- **info for** requires the full file name with extension
- **path to** will report the full app name with extension
- We'll provide more information about extensions in future releases

Paths and Extensions

- Script Editor opens files with extensions only
  - Compiled script files: **.scpt**
  - Uncompiled AppleScript text: **.applescript**
  - Terminology: **.osax, .asdictionary**
- For now, the user must add extensions if desired
- Script Editor continues to set type and creator on new scripts and applets

What We're Working On

- Scriptability of Apple applications
- Networked Apple events for Mac OS X
- Smoother integration with features of Mac OS X
- Plug-in model for applications and AppleScript
- Object Model support for lists, records, strings

Summary

- AppleScript in Mac OS X is very usable
- We're continuing to add new stuff
  - Unicode support
- Mac OS X and Classic versions will stay in synch for a while
- The Mac OS X version will support new features and idiosyncracies of the OS

Roadmap

134 Building AppleScript Applications: The nitty-gritty of development on Mac OS X	Here! **Next!**
FF115 AppleScript Feedback Forum Let us know how we're doing	Room C **Fri., 5:00 p.m.**

Who to Contact

Jason Yeo
Mac OS Technology Manager
Apple Developer Group
jason@apple.com

Chris Espinosa
Manager, Components & Scripting
Apple Software Engineering
cde@apple.com

Sal Soghoian
AppleScript Marketing Manager
Apple Worldwide Product Marketing
sal@apple.com

http://developer.apple.com/events/urls.html

Q&A

With the AppleScript Engineering Team

http://developer.apple.com/events/urls.html

WWDC
Worldwide Developers Conference 2001